US-PAT-NO: 6338122

DOCUMENT-IDENTIFIER: US 6338122 B1

TITLE: Non-uniform memory access (NUMA) data processing system that speculatively forwards a read request to a remote processing node

DATE-ISSUED: January 8, 2002

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Baumgartner; Yoanna	Austin	TX	N/A	N/A
Dean; Mark Edward	Austin	TX	N/A	N/A
Elman; Anna	Austin	TX	N/A	N/A

US-CL-CURRENT: 711/141, 711/100 , 711/124 , 711/147 , 711/154

ABSTRACT:

A non-uniform memory access (NUMA) computer system includes at least a local processing node and a remote processing node that are each coupled to a node interconnect. The local processing node includes a local interconnect, a processor and a system memory coupled to the local interconnect, and a node controller interposed between the local interconnect and the node interconnect. In response to receipt of a read request from the local interconnect, the node controller speculatively transmits the read request to the remote processing node via the node interconnect. Thereafter, in response to receipt of a response to the read request from the remote processing node, the node controller handles the response in accordance with a resolution of the read request at the local processing node. For example, in one processing scenario, data contained in the response received from the remote processing node is discarded by the node controller if the read request received a Modified Intervention **coherency** response at the local processing node.

24 Claims, 8 Drawing figures

Exemplary Claim Number: 10

Number of Drawing Sheets: 8

---------- KWIC ---------

Abstract Text - ABTX (1):

A non-uniform memory access (NUMA) computer system includes at least a local processing node and a remote processing node that are each coupled to a node interconnect. The local processing node includes a local interconnect, a processor and a system memory coupled to the local interconnect, and a node controller interposed between the local interconnect and the node interconnect. In response to receipt of a read request from the local interconnect, the node controller speculatively transmits the read request to the remote processing node via the node interconnect. Thereafter, in response to receipt of a response to the read request from the remote processing node, the node controller handles the response in accordance with a resolution of the read request at the local processing node. For example, in one processing scenario, data contained in the response received from the remote processing node is discarded by the node controller if the read request received a Modified Intervention **coherency** response at the local processing node.

Brief Summary Text - BSTX (7):

As a result, an MP computer system topology known as non-uniform memory access (NUMA) has emerged as an alternative design that addresses many of the limitations of SMP computer systems at the expense of some additional complexity. A typical NUMA computer system includes a number of interconnected nodes that each include one or more processors and a local "system" memory. Such computer systems are said to have a non-uniform memory access because each processor has lower access latency with respect to data stored in the system memory at its local node than with respect to data stored in the system memory at a remote node. NUMA systems can be further classified as either

non-**coherent** or cache **coherent,** depending upon whether or not data **coherency** is maintained between caches in different nodes. The complexity of cache **coherent** NUMA (CC-NUMA) systems is attributable in large measure to the additional communication required for hardware to maintain data **coherency** not only between the various levels of cache memory and system memory within each node but also between cache and system memories in different nodes. NUMA computer systems do, however, address the scalability limitations of conventional SMP computer systems since each node within a NUMA computer system can be implemented as a smaller SMP system. Thus, the shared components within each node can be optimized for use by only a few processors, while the overall system benefits from the availability of larger scale parallelism while maintaining relatively low latency.

Brief Summary Text - BSTX (10):

In accordance with the present invention, a non-uniform memory access (NUMA) computer system includes at least a local processing node and a remote processing node that are each coupled to a node interconnect. The local processing node includes a local interconnect, a processor and a system memory coupled to the local interconnect, and a node controller interposed between the local interconnect and the node interconnect. In response to receipt of a read request from the local interconnect, the node controller speculatively transmits the read request to the remote processing node via the node interconnect. Thereafter, in response to receipt of a response to the read request from the remote processing node, the node controller handles the response in accordance with a resolution of the read request at the local processing node. For example, in one processing scenario, data contained in the response received from the remote processing node is discarded by the node controller if the read request received a Modified Intervention **coherency** response at the local processing node.

Detailed Description Text - DETX (4):

Each of processing nodes 8a-8n further includes a respective node

controller
20 coupled between local interconnect 16 and node interconnect 22.
Each node
controller 20 serves as a local agent for remote processing nodes 8 by
performing at least two functions. First, each node controller 20
snoops the
associated local interconnect 16 and facilitates the transmission of
local
communication transactions to remote processing nodes 8. Second, each
node
controller 20 **snoops** communication transactions on node interconnect 22
and
masters relevant communication transactions on the associated local
interconnect 16. Communication on each local interconnect 16 is
controlled by
an arbiter 24. Arbiters 24 regulate access to local interconnects 16
based on
bus request signals generated by processors 10 and compile **coherency**
responses
for **snooped** communication transactions on local interconnects 16, as
discussed
further below.

Detailed Description Text - DETX (8):
For purposes of the present discussion, the processing node 8 that
stores a
particular datum in its system memory 18 is said to be the **home node**
for that
datum; conversely, others of processing nodes 8a-8n are said to be
remote nodes
with respect to the particular datum.

Detailed Description Text - DETX (9):
Memory **Coherency**

Detailed Description Text - DETX (10):
Because data stored within each system memory 18 can be requested,
accessed,
and modified by any processor 10 within NUMA computer system 6, NUMA
computer
system 6 implements a cache **coherence** protocol to maintain **coherence**
both
between caches in the same processing node and between caches in
different
processing nodes. Thus, NUMA computer system 6 is properly classified
as a
CC-NUMA computer system. The cache **coherence** protocol that is
implemented is
implementation-dependent and may comprise, for example, the well-known
Modified, Exclusive, Shared, Invalid (**MESI**) protocol or a variant
thereof.
Hereafter, it will be assumed that cache hierarchies 14 and arbiters 24
implement the conventional **MESI** protocol, of which node controllers 20

recognize the M, S and I states and consider the E state to be merged into the M state for correctness. That is, node controllers 20 assume that data held exclusively by a remote cache has been modified, whether or not the data has actually been modified.

Detailed Description Text - DETX (12):

Local interconnects 16 and node interconnect 22 can each be implemented with any bus-based broadcast architecture, switch-based broadcast architecture, or switch-based non-broadcast architecture. However, in a preferred embodiment, at least node interconnect 22 is implemented as a switch-based non-broadcast interconnect governed by the 6xx communication protocol developed by IBM Corporation. Local interconnects 16 and node interconnect 22 permit split transactions, meaning that no fixed timing relationship exists between the address and data tenures comprising a communication transaction and that data packets can be ordered differently than the associated address packets. The utilization of local interconnects 16 and node interconnect 22 is also preferably enhanced by pipelining communication transactions, which permits a subsequent communication transaction to be sourced prior to the master of a previous communication transaction receiving **coherency** responses from each recipient.

Detailed Description Text - DETX (13):

Regardless of the type or types of interconnect architecture that are implemented, at least three types of "packets" (packet being used here generically to refer to a discrete unit of information)--address, data, and **coherency** response--are utilized to convey information between processing nodes 8 via node interconnect 22 and between **snoopers** via local interconnects 16. Referring now to Tables I and II, a summary of relevant fields and definitions are given for address and data packets, respectively.

Detailed Description Text - DETX (14):

As indicated in Tables I and II, to permit a recipient node or **snooper** to

determine the communication transaction to which each packet belongs, each packet in a communication transaction is identified with a transaction tag. Those skilled in the art will appreciate that additional flow control logic and associated flow control signals may be utilized to regulate the utilization of the finite communication resources.

Detailed Description Text - DETX (15):

Within each processing node 8, status and **coherency** responses are communicated between each **snooper** and the local arbiter 24. The signal lines within local interconnects 16 that are utilized for status and **coherency** communication are summarized below in Table III.

Detailed Description Text - DETX (16):

Status and **coherency** responses transmitted via the AResp and AStat lines of local interconnects 16 preferably have a fixed but programmable timing relationship with the associated address packets. For example, the AStatOut votes, which provide a preliminary indication of whether or not each **snooper** has successfully received an address packet transmitted on local interconnect 16, may be required in the second cycle following receipt of the address packet. Arbiter 24 compiles the AStatOut votes and then issues the AStatIn vote a fixed but programmable number of cycles later (e.g., 1 cycle). Possible AStat votes are summarized below in Table IV.

Detailed Description Text - DETX (17):

Following the AStatIn period, the ARespOut votes may then be required a fixed but programmable number of cycles (e.g., 2 cycles) later. Arbiter 24 also compiles the ARespOut votes of each **snooper** and delivers an ARespIn vote, preferably during the next cycle. The possible AResp votes preferably include the **coherency** responses listed in Table V.

Detailed Description Text - DETX (18):

The ReRun AResp vote, which is usually issued by a node controller 20, indicates that the **snooped** request has a long latency and that the source of

the request will be instructed to reissue the transaction at a later time.
Thus, in contrast to a Retry AResp vote, a ReRun makes the recipient of a
transaction that voted ReRun (and not the originator of the transaction)
responsible for causing the communication transaction to be reissued at a later
time.

Detailed Description Text - DETX (20):
Referring now to FIG. 2, there is illustrated a more detailed block diagram
of a node controller 20 in NUMA computer system 6 of FIG. 1. As shown in FIG.
2, each node controller 20, which is coupled between a local interconnect 16
and node interconnect 22, includes a transaction receive unit (TRU) 40, a
transaction send unit (TSU) 42, a data receive unit (DRU) 44, and a data send
unit (DSU) 46. TRU 40, TSU 42, DRU 44 and DSU 46 can be implemented, for
example, with field programmable gate arrays (FPGAs) or application specific
integrated circuits (ASICs). As indicated, the address and data paths through
node controller 20 are bifurcated, with address (and **coherency**) packets being
processed by TRU 40 and TSU 42 and data packets being processed by DSU 44 and
DRU 46.

Detailed Description Text - DETX (21):
TRU 40, which is so designated to indicate transaction flow off of node
interconnect 22, is responsible for accepting address and **coherency** packets
from node interconnect 22, issuing transactions on local interconnect 16, and
forwarding responses to TSU 42. TRU 40 includes response multiplexer (mux) 52,
which receives packets from node interconnect 22 and passes selected packets to
both bus master 54 and **coherency** response logic 56 within TSU 42. In response
to receipt of a address packet from response multiplexer 52, bus master 54 can
initiate a communication transaction on its local interconnect 16 that is the
same as or different from the type of communication transaction indicated by
the received address packet.

Detailed Description Text - DETX (22):

TSU 42, which as indicated by its nomenclature is a conduit for transactions flowing onto node interconnect 22, includes a multiple-entry pending buffer 60 that temporarily stores attributes of communication transactions sourced onto node interconnect 22 that have yet to be completed. The transaction attributes stored in an entry of pending buffer 60 preferably include at least the address (including tag) of the transaction, the type of the transaction, and the number of expected **coherency** responses. Each pending buffer entry has an associated status, which can be set either to Null, indicating that the pending buffer entry can be deleted, or to ReRun, indicating that the transaction is still pending. In addition to sourcing address packets on node interconnect 22, TSU 42 interacts with TRU 40 to process memory request transactions and issues commands to DRU 44 and DSU 46 to control the transfer of data between local interconnect 16 and node interconnect 22. TSU 42 also implements the selected (i.e., MSI) **coherency** protocol for node interconnect 22 with **coherency** response logic 56 and maintains **coherence** directory 50 with directory control logic 58.

Detailed Description Text - DETX (23):

Coherence directory 50 stores indications of the system memory addresses of data (e.g., cache lines) checked out to caches in remote nodes for which the local processing node is the **home node**. The address indication for each cache line is stored in association with an identifier of each remote processing node having a copy of the cache line and the **coherency** status of the cache line at each such remote processing node. Possible **coherency** states for entries in **coherency** directory 50 are summarized in Table VI.

Detailed Description Text - DETX (24):

As indicated in Table VI, the knowledge of the **coherency** states of cache lines held by remote processing nodes is imprecise. This imprecision is due to the fact that a cache line held remotely can make a transition from S

to I,
from E to I, or from E to M without notifying the node controller 20 of the
home node.

Detailed Description Text - DETX (26):

Referring now to FIGS. 3A and 3B, there are illustrated two high level logical flowcharts that together depict an exemplary method for processing read request transactions in accordance with the present invention. Referring first to FIG. 3A, the process begins at block 70 and thereafter proceeds to block 72, which depicts a processor 10, such as processor 10a of processing node 8a, issuing a read request transaction on its local interconnect 16. The read request transaction is received by node controller 20 and the rest of the **snoopers** coupled to local interconnect 16 of processing node 8a. In response to receipt of the read request, the **snoopers** drive AStatOut votes, which are compiled by arbiter 24 to generate an AStatIn vote, as shown at block 74. Before node controller 20 supplies an Ack AStatOut vote to permit the read request to proceed, node controller 20 allocates both a read entry and write-with-clean entry in pending buffer 60, if the read request specifies an address in a remote system memory 18. As discussed further below, by allocating both entries, node controller 20 is able to speculatively forward the read request to the **home node** of the requested cache line and correctly handle the response to the read request regardless of the outcome of the subsequent AResp vote at processing node 8a.

Detailed Description Text - DETX (27):

Referring now to block 76, if the AStatIn vote generated at block 74 is Retry, the read request is essentially killed, allocated entries, if any, in pending buffer 60 are freed, and the process returns to block 72, which has been described. In this case, processor 10a must reissue the read request at a later time. If, on the other hand, the AStatIn vote generated at block 74 is not Retry, the process proceeds from block 76 to block 78, which depicts node controller 20 determining by reference to the memory map whether or not

its
processing node 8 is the **home node** of the physical address specified in the
read request. If so, the process proceeds to block 80; however, if the local
processing node 8 is not the **home node** for the read request, the process
proceeds to block 100.

Detailed Description Text - DETX (28):

Referring now to block 80, the **snoopers** within processing node 8a then
provide their ARespOut votes, which arbiter 24 compiles to generate an ARespIn
vote. If **coherency** directory 50 indicates that the cache line identified by
the address specified in the read request is checked out to at least one remote
processing node 8, node controller 20 will vote ReRun if servicing the read
request requires communication with a remote processing node 8. For example,
if **coherency** directory 50 indicates that a requested cache line is Modified at
a remote processing node 8, servicing a read request will entail forwarding the
read request to the remote processing node 8. Similarly, if **coherency**
directory 50 indicates that a requested cache line is Shared at a remote
processing node 8, servicing a read-with-intent-to-modify (RWITM) request will
entail transmitting a Kill command to the remote processing node 8 to
invalidate the remote copy or copies of the requested cache line. As shown at
block 82, if the ARespIn vote is not ReRun, the process passes to block 90,
which is described below; if the ARespIn vote is ReRun, the process proceeds to
block 84.

Detailed Description Text - DETX (29):

Block 84 illustrates node controller 20 transmitting, via node interconnect
22, an appropriate transaction to the one or more remote processing nodes 8
that have checked out the requested cache line. As noted above, the
transaction may be either a cache command (e.g., Kill) or a read request
transaction. The process then iterates at block 86 until a response is
received by node controller 20 from each remote processing node 8 to which a
transaction was transmitted at block 84. Following receipt of the appropriate
number of responses, which may include the receipt of a copy of the

requested
cache line, node controller 20 transmits a ReRun request on local
interconnect
16, instructing requesting processor 10a to reissue the read request.
As
indicated at block 88, requesting processor 10a responds to the ReRun
request
by reissuing the read request transaction on local interconnect 16.
Following
the AStat and AResp periods, the read request is serviced at block 90,
either
by node controller 20 supplying a copy of the requested cache line
received
from a remote processing node 8 or by another local **snooper** in
processing node
8a (e.g., memory controller 17 or a cache hierarchy 14) sourcing the
requested
cache line. Thereafter, the process terminates at block 150.

Detailed Description Text - DETX (30):
   Referring now to block 100, if node controller 20 of processing node
8a
determines that processing node 8a is not the **home node** for the
requested cache
line, node controller 20 speculatively forwards the read request
transaction to
the remote processing node 8 that is the **home node** for the requested
cache
line. As indicated in FIG. 3A, the read request is forwarded by node
controller 20 at least concurrently with the ARespIn period and is
preferably
forwarded immediately following receipt of the AStatIn vote from
arbiter 24 and
prior to the ARespOut period. When the read request is forwarded, the
status
of the read entry in pending buffer 60 is updated to ReRun. Then, as
shown at
block 102, the **snoopers** provide their ARespOut votes, which arbiter 24
compiles
to generate a ARespIn vote. Thereafter, as illustrated at block 110
and
following blocks, the **home node** supplies a response to the read
request, and
node controller 20 handles the response in accordance with the ARespIn
vote for
the read request at processing node 8a.

Detailed Description Text - DETX (31):
   If the ARespIn vote is Retry, the read request is essentially killed
at
processing node 8a. Thus, in response to a receipt of a ARespIn Retry
vote,
the status of the read and write entries allocated in pending buffer 60
are

updated to Null. The process passes then through block 110 to blocks 112 and 114, which depict node controller 20 waiting to receive the requested cache line from the **home node** and discarding the cache line when received in response to the Null status of the read entry in pending buffer 60. The process then terminates at block 150.

Detailed Description Text - DETX (32):
If the ARespIn vote is Modified Intervention, the read request can be serviced locally at processing node 8a without utilizing (stale) data from the **home node**. Thus, in response to a receipt of a ARespIn Modified Intervention vote, the status of the read entry in pending queue 60 is updated to Null, and the process proceeds from block 102 through blocks 110 and 120 to block 122. Block 122 illustrates the **snooper** that voted Modified Intervention during the ARespOut period sourcing the requested cache line on local interconnect 16 of processing node 8a. The **coherency** state of the requested cache line at the **snooper** sourcing the requested cache line is then updated from Modified to Shared. In response to receiving the requested cache line, requesting processor 10a loads the requested cache line into its cache hierarchy 14, as illustrated at block 124. In addition, node controller 20 captures the requested cache line off of local interconnect 16 and issues a write-with-clean transaction containing the cache line to the **home node** in order to update the **home node's** system memory 18 with the modified cache line, as depicted at block 126. The process then passes to block 112, which has been described.

Detailed Description Text - DETX (33):
The **coherence** protocol implemented by computer system 6 may optionally support shared intervention, that is, the servicing of a read request transaction by a local cache hierarchy 14 that holds the requested cache line in Shared state. If shared intervention is supported by the cache **coherence** protocol of computer system 6 and the ARespIn vote for the request transaction is Shared (i.e., Shared Intervention), the **snooper** voting Shared sources the requested cache line on local interconnect 16, as depicted at block

132. In response to receiving the requested cache line, requesting processor 10a loads the requested cache line into its cache hierarchy 14, as illustrated at block 134. As no update to system memory 18 is required, the status of the read and write entries allocated in pending buffer 60 are updated to Null, and the process terminates at block 150.

Detailed Description Text - DETX (34):
Finally, if the ARespIn vote for the request transaction at processing node 8a is ReRun, the status of the write entry in pending buffer 60 is updated to Null and that of the read entry is set to ReRun. The process then proceeds from block 102 through blocks 110, 120, 130 to block 142, which depicts node controller 20 of processing node 8a waiting until the requested cache line is received from the **home node**. In response to receipt of the requested cache line from the **home node** via node interconnect 22, node controller 20 transmits the requested cache line to requesting processor 10a via local interconnect 16, as shown at block 144. In response to receipt of the requested cache line, requesting processor 10a loads the requested cache line into its cache hierarchy 14, as illustrated at block 146. The process then terminates at block 150.

Detailed Description Text - DETX (35):
Referring now to FIG. 3B, there is depicted a high level logical flowchart illustrating how the **home node** processes a transaction received from another processing node. As illustrated, the process begins at block 160 and thereafter proceeds to block 162, which illustrates a determination of whether or not the **home node** has received a transaction from another processing node via node interconnect 22. If not, the process simply iterates at block 162 until a transaction is received from another processing node 8. In response to receipt by the **home node's** node controller 20 of a transaction from a remote processing node 8, the process passes to block 164, which depicts the **home node's** node controller 20 transmitting the transaction received at

block 162 on
the local interconnect 16 of the **home node**. As indicated by decision block
170, if the transaction issued on local interconnect 16 is a read transaction,
the process proceeds to block 172, which illustrates the read request being
serviced by a **snooper** that supplies a copy of the requested cache line to the
home node's node controller 20. In response to receipt of the requested cache
line, node controller 20 transmits the requested cache line to the requesting
processing node 8 via node interconnect 22, as depicted at block 174.
Thereafter, the process terminates at block 190.

Detailed Description Text - DETX (36):
Returning to block 164, if the transaction transmitted on the **home node's**
local interconnect 16 is a write (e.g., write-with-clean) transaction, the
process proceeds through blocks 170 and 180 to block 184, which illustrates
memory controller 17 updating system memory 18 with the cache line contained in
the write transaction. The process then terminates at block 190. If the
transaction transmitted on the **home node's** local interconnect 16 is neither a
read transaction nor a write transaction, the **home node** performs the action(s)
indicated by the transaction at block 182, and the process terminates at block
190. The actions that may be performed in response to a transaction other than
a read or write transaction include, for example, updates to the **coherence**
states of cache lines held in the **home node's** cache hierarchies 14.

Detailed Description Text - DETX (37):
Referring now to FIGS. 4A-4D, there is depicted an exemplary processing
scenario in accordance with the present invention. For clarity, the exemplary
processing scenario is explained below utilizing a simplified representation of
computer system 6 having two processing nodes 8a and 8b, which each contain two
processors 10a and 10b. The **coherence** state of the requested cache line is
indicated within the cache hierarchy 14 of each processor 10 and within
coherence directory 50 of **home node** 8a.

Detailed Description Text - DETX (38):
As indicated in FIG. 4A, processor 10b of processing node 8b first issues a read request for a cache line that is Invalid (i.e., not resident) in its cache hierarchy 14. In response to receiving the read **request, node** controller 20 of processing node 8b speculatively transmits the read request to processing node 8a, which is the **home node** of the cache line specified in the read request. After the read request is speculatively forwarded to processing node 8a, processor 10a votes Modified Intervention during the ARespOut period because its cache hierarchy 14 holds the requested cache line in Modified state. The arbiter of processing node 8b compiles the ARespOut votes and supplies a Modified Intervention ARespIn vote to each **snooper** in processing node 8b.

Detailed Description Text - DETX (39):
Next, as shown in FIG. 4B, node controller 20 of processing node 8a receives the speculatively forwarded read request and issues the read request on its local interconnect 16. As indicated in FIG. 4B, node controller 20 votes Null during the ARespOut period in response to **coherence** directory 50 indicating that the cache line specified in the read request is Modified at processing node 8b. Node controller 20 recognizing this special condition permits the read request to proceed, as discussed below with respect to FIG. 4D.

Detailed Description Text - DETX (40):
As illustrated in FIG. 4C, independently of (and possibly prior to, concurrently with, or after) the speculative forwarding of the read request to processing node 8a, processor 10a of processing node 8b responds to the read request by sourcing the requested cache line on local interconnect 16 and updating the **coherence** state of the requested cache line in its cache hierarchy 14 to Shared. In response to **snooping** the requested cache line, requesting processor 10b loads the requested cache line into its cache hierarchy 14 and sets the associated **coherence** state to Shared. In addition, node controller 20 of processing node 8b captures the cache line and issues a

write-with-clean
transaction containing the modified cache line to processing node 8a.
In
response to receipt of the write-with-clean transaction, node
controller 20 of
processing node 8a issues the write-with-clean to system memory 18 via
its
local interconnect 16. System memory 18 of **home node** 8a then updates
the
corresponding memory line with the modified data.

Detailed Description Text - DETX (42):
As has been described, the present invention provides an improved
NUMA
computer system and an improved communication methodology in a NUMA
computer
system. In accordance with the present invention, a read request
transaction
is speculatively issued to a remote (i.e., home) processing node via
the node
interconnect prior to a determination of whether the read request can
be
serviced locally without the intervention of the remote processing
node. When
the remote processing node responds to the speculatively forwarded read
request, the requesting processing node handles the response in
accordance with
the local **coherence** response for the read request. In this manner, the
latency
of communication transactions can be dramatically reduced.

Detailed Description Paragraph Table - DETL (1):
TABLE I Field Name Description Address Modifiers defining
attributes of a
<0:7> communication transaction for **coherency,** write thru, and
protection Address Tag used to identify all packets within a
<8:15>
communication transaction Address Address portion that indicates the
<16:63> physical, virtual or I/O address in a request Aparity
Indicates
parity for address bits <0:63> <0:2> TDescriptors
Indicate size
and type of communication transaction

Detailed Description Paragraph Table - DETL (2):
TABLE I Field Name Description Address Modifiers defining
attributes of a
<0:7> communication transaction for **coherency,** write thru, and
protection Address Tag used to identify all packets within a
<8:15>
communication transaction Address Address portion that indicates the
<16:63> physical, virtual or I/O address in a request Aparity
Indicates

parity for address bits <0:63> <0:2> TDescriptors
Indicate size
and type of communication transaction

Detailed Description Paragraph Table - DETL (3):
TABLE III Signal Name Description AStatOut Encoded signals
asserted by
each bus <0:1> receiver to indicate flow control or error
information
to arbiter AStatIn Encoded signals asserted by arbiter in <0:1>
response to tallying the AStatOut signals asserted by the bus
receivers
ARespOut Encoded signals asserted by each bus <0:2> receiver to
indicate
coherency information to arbiter ARespIn Encoded signals asserted by
arbiter
in <0:2> response to tallying the ARespOut signals asserted by
the bus
receivers

Detailed Description Paragraph Table - DETL (4):
TABLE IV AStat vote Meaning Null Idle Ack Transaction accepted
by
snooper Error Parity error detected in transaction Retry Retry
transaction,
usually for flow control

Detailed Description Paragraph Table - DETL (5):
TABLE V **Coherency** responses Meaning Retry Source of request must
retry
transaction usually for flow control reasons Modified Line is
modified in
cache and will be intervention sourced to requestor Shared Line is
held
shared in cache Null Line is invalid in cache ReRun **Snooped** request
has long
latency and source of request will be instructed to reissue
transaction at a
later time

Detailed Description Paragraph Table - DETL (6):
TABLE VI Possible Possible state(s) **Coherence** state(s) in
directory in
local remote state cache cache Meaning Modified I M, E, or Cache line
may be
(M) I modified at a remote node with respect to system memory at home
node
Shared S or I S or I Cache line may be held (S) non-exclusively at
remote
node Invalid M, E, S, I Cache line is not held (I) or I by any remote
node
Pending- S or I S or I Cache line is in the shared process of being

invalidated at remote nodes Pending- I M, E, or Cache line, which may
modified I be modified remotely, is in process of being written back to
system memory at **home node,** possibly with invalidation at remote node

Claims Text - CLTX (3):
at least a local node and a remote node that are each coupled to said node interconnect, said local node including a local interconnect and said local and remote nodes each including one or more **snoopers,** wherein said one or more **snoopers** of said local node include a node controller interposed between said local interconnect and said node interconnect, wherein said node controller speculatively transmits a request transaction received from said local interconnect to said remote node via said node interconnect prior to determination of a local node response, wherein **snoopers** at the local node and the remote node independently process the request transaction to obtain the local node response at the local node and a remote node response at the remote node, and wherein said node controller of the local node handles the remote node response to said request transaction in accordance with the local node response.

Claims Text - CLTX (10):
5. The computer system of claim 4, wherein said node controller of said local node discards data received from said remote node in response to said request transaction if said local node response includes a modified or shared intervention **coherency** response.

Claims Text - CLTX (13):
8. The computer system of claim 1, wherein **snoopers** at said local node and said remote node process said request transaction with independent timing.

Claims Text - CLTX (15):
10. A method of operation in a computer system including a node interconnect that couples at least a local node and a remote node that each contain one or more **snoopers,** wherein said one or more **snoopers** at said

local
node include a node controller interposed between a local interconnect of the
local node and said node interconnect, said method comprising:

Claims Text - CLTX (17):
independently processing the request transaction by **snoopers** of the local
node and **snoopers** of the remote node to obtain the local node response at the
local node and a remote node response at the remote node;

Claims Text - CLTX (26):
15. The method of claim 10, wherein handling said response comprises
sourcing data received from said remote processing node onto said local
interconnect of said local processing node if said request transaction receives
a **coherency** response at said local processing node indicating that said request
transaction cannot be serviced locally.

Claims Text - CLTX (27):
16. The method of claim 10, wherein **snoopers** at said local node and said
remote node process said request transaction with independent timing.

Claims Text - CLTX (29):
18. The method of claim 17, wherein handling said remote node response
comprises discarding data received from said remote node if said local node
response includes a modified or shared intervention **coherency** response.

Claims Text - CLTX (32):
one or more **snoopers** coupled to the local interconnect, wherein said one or
more **snoopers** include a node controller providing an interface for a node
interconnect for interconnecting multiple nodes, wherein said node controller
speculatively transmits a request transaction received from said local
interconnect to the node interconnect prior to determination of a local node
response by the one or more **snoopers**, and wherein the node controller handles a
remote node response to said request transaction received from the node
interconnect in accordance with the local node response.

Claims Text - CLTX (39):

23. The node of claim 22, wherein said node controller discards data
received from said node interconnect in response to said request transaction if
said local node response includes a modified or shared intervention **coherency**
response.